# INTRODUCTION TO COMPLEX FAULT PROTECTION SOFTWARE TESTING

Sue A. Johnson
Jet Propulsion Laboratory, California Institute of Technology
M.S. 198-219, 4800 Oak Grove Drive
Pasadena, California 91109-8099

## ABSTRACT

This paper describes how fault protection software verification testing has been addressed for the Attitude and Articulation Control Subsystem on the Saturn-bound Cassini spacecraft. The verification included definition of failure injection capabilities to the test beds. The Cassini Fault Protection test program evolved using several test phases to complete separate goals which taken together encompassed all of the flight software complexities. This phased testing approach developed high confidence in the robustness and correctness of the FP flight software. Over 500 prelaunch tests of the flight software in a realistically modeled environment were used to ensure the design is robust to single faults. In the process we found that many double fault scenarios are handled by the Cassini Attitude and Articulation Control Subsystem fault protection software design.

## 1. SCOPE OF CASSINI AACS FAULT PROTECTION SOFTWARE VERIFICATION

The Saturn-bound Cassini spacecraft has the most advanced Attitude and Articulation Control Subsystem (AACS) fault protection design ever attempted on any interplanetary spacecraft. The system that the AACS fault protection oversees is dual redundant. Within the AACS are two 1750A microprocessor based flight computers, two MIL-STD-1553B type communications buses, and sixteen remote terminals connected to eight different types of hardware (e.g. sun sensors). The flight computers, buses, and remote terminals are all cross-strapped for full redundancy.

The Cassini AACS flight software is complex as well. The object oriented flight software, written in Ada, has nineteen operational modes. The number of elements in the AACS fault protection is overwhelming: 312 error monitors, 310 activation rules, and 214 response scripts. The AACS fault protection design document is four inches thick. There are three other subsystems to which AACS has interfaces: Command and Data Subsystem (CDS), Power, and Propulsion.

Fault Protection verification progress was tracked by use of Excel spreadsheets and weekly test result reviews. The reviews were attended by the members of the fault protection test team, the fault protection designers, a software test bed representative, and a flight software representative. At these weekly meetings the apparently successful tests would be passed along for an independent verification. All test problems were assigned to the likely culprit, test bed error or flight software error. Flight software problems were described in detail as a Flight Software Change (FSC) request. All test results obtained using the modified flight software were reported to the weekly test review group to close the loop.

The AACS fault protection weekly review meetings lasted about 1.5 hours and could summarize about 25 tests per meeting. The fault protection verification team included 2.5 full time equivalent people for testing. The fault protection design team had 2.0 full time equivalents spread over 4 to 5 people. The flight software representative spent about 40% of her week on fault protection related issues.

The fault protection verification program had only one year to prove the AACS fault protection design. Testing began in September, 1996 with the goal of verifying all response scripts by mid-December. This initial goal would support later system level testing on the partially assembled spacecraft beginning in January, 1997. Other drivers of the AACS fault protection verification schedule were the May, 1997 shipping date to Kennedy Space Center, and the scheduled delivery of the launch AACS flight software in June 1997.

## 2. FAULT PROTECTION TEST BEDS

Cassini's AACS has two useful test beds for fault protection test verification. The highest fidelity test bed is hardware based. In this laboratory the flight computers run the flight software and interface with flight spares, protoflight boards or engineering models. Sun and stars are always simulated. The spacecraft dynamics are modeled. The second fault protection verification test bed is based on a 1750A emulation on Sun workstations. This software based test bed uses the same dynamics models, and star emulation. All the AACS hardware have detailed representative models in the software test bed.

In addition to faithfully representing the AACS, these test beds provide fault injection capabilities. Potential hardware failures were captured in Failure Modes and Effects Analysis (FMEA) reports. A significant effort was expended in defining failure descriptions and associated symptoms from the FMEA. The output was the wish list of failure injection capabilities for the two test beds, see Table 1. Not all failure injections could be implemented in both labs. But most could be implemented in at least one of the labs.

Table 1. Comparisons of Failure Injection Wish List to Delivered Capabilities

| Failure Location | # of Wish List failures | Software Test Bed | Hardware Test Bed Capabilities |
|---|---|---|---|
| Sun Sensors | 17 | 16 | 4 |
| Inertial Reference Units | 21 | 21 | 7 |
| Solid State Power | 14 | 13 | 11 |
| Accelerometer | 17 | 17 | 3 |
| Engine Gimbal Assembly | 27 | 26 | 5 |
| Reaction Wheels | 14 | 14 | 3 |
| Stellar Reference Units | 19 | 14 | 3 |
| Bi-Prop System | 79 | 79 | 25 |
| Mono-Prop System | 41 | 41 | 19 |
| Attitude Control | 15 | 15 | 3 |
| XBA & CDS Messages | 91 | 63 | 28 |
| Remote Terminal & Bus | 52 | 51 | 5 |
| Star Identification | 10 | 7 | 1 |
| Attitude Estimation | 8 | 8 | 1 |
| Totals | 425 | 385 | 118 |

There are strong advantages of testing in the hardware based test bed. The system is more flight-like than the pure software approach. The bus transactions have real timing. The interplay between the two AACS flight computers is testable, including flight computer reset recovery. The hardware test bed disadvantages include: non-CDS synchronized clocks, real-time speed, less visibility into software states, and longer set-up time. The support software is easily swamped by quickly changing dynamic situations. This test bed was also in high demand for other AACS functional testing. Fault protection verification test time was very constrained. The fault injection capability is limited to around 100 types of failures.

The software test bed bore the majority of the test load since multiple test bed versions could be spawned concurrently by multiple users, and since it could be run over night or weekends unsupervised. It runs on the order of 4 to 10 times faster than real-time. Flight software variables can be investigated without interruption of the test. Four sets of mass properties are available to simulate different phases of the Cassini mission. There are nearly 400 types of fault injection capabilities in the software test bed. The software test bed disadvantages include its unlimited processing power, which can mask timing issues. Flight software exceptions, a fatal error on the Sun Ada flight software versions, would occur due to counter roll over. The software test bed could not simulate a flight computer reset and recovery or a CDS to AACS communications loss.

## 3. TEST PHASES

The Cassini AACS fault protection test program used several test phases to complete separate goals which taken together encompassed all of the flight software complexities. The goal of the first phase was to test all response scripts, the code that initiates remedial actions to a sensed failure, at least once. In accomplishing this phase over half the error monitors were tested as well. Phase one began in September 1996 and finished at the end of December 1996. The goal of the second phase was to verify AACS's fault protection interfaces with the System fault protection. Cassini's System fault protection provides responses for spacecraft Safing, Command Loss, AACS Heartbeat Loss, Over-temperature, Over-pressure and Under-voltage. Phase two lasted three weeks. The goal of the third phase was to finish the error monitor verification. Phase three stretched from January 1997 to April 1997. The fourth phase of testing exercised every path for the 27 response scripts with more than two paths. Path testing lasted from April 1997 through early June 1997. The fifth phase worked on stress testing the fault protection software, usually by injecting multiple faults. Fault protection verification testing continued up through early October 1997. In addition to the five phases of focused testing, fault protection requirements were verified throughout the test program.

AACS fault protection verification regression testing was performed in the two weeks prior to any AACS flight software delivery to the spacecraft assembly team. Previous test cases that could span the possible hardware and functional failures were selected for regression testing. These tests produced only two instances of new a problem.

## 4. TEST RESULTS

Throughout the AACS fault protection verification period, types of test failures were tracked. Test failures were assigned to three categories: test script problem, test bed problem, flight software problem. The flight software problem category includes code errors, software design errors, and even hardware design errors. Table 2 below summarizes the test failures by category.

Table 2: Test Failure Distribution

| Test Failure Source | % |
|---|---|
| Script | 15 |
| Test Bed | 25 |
| Flight Software | 60 |

AACS fault protection verification tests for the software test bed were written as Tcl, tool command language, scripts. [1] The script controls the time explicitly. The time required to write a new test script decreased from 40 minutes to 10 minutes for a moderately complex test as our familiarity with the approach increased. Some of the most common mistakes that would require a re-run were not capturing appropriate data, using incompatible sun position and ephemeris data, and forgetting an enabling command to set up the AACS system to the expected conditions. A good example of an enabling command problem is trying to switch from the branch A thruster control system to branch B. The commands to do so entail powering on the monopropellant driver on branch B, opening the branch B latch valve, and warming the branch B catalyst bed heaters. After these steps are accomplished, commands may be sent to swap thrusters. Thereafter, the hardware on branch A may be closed and powered off. The commonly missed step was opening the latch valve. There were many other similar command sequence issues that must be addressed in designing the test scripts.

In the hardware test bed the scripts were written as an overlay sequence. Once the flight computer began running time was advancing. The scripts usually issued a few commands, then waited for telemetry confirmation before continuing. The test errors in this laboratory include those listed above, but also suffered from the state the hardware had been left in by the previous user.

Test bed functional problems were similar in both the hardware and software test beds. Sometimes the test bed would crash. Sometimes the injected failure would do nothing. Sometimes a hardware model would behave in a suspicious manner. An example of a hardware model problem comes from the hardware test bed. The two Cassini AACS inertial reference units each have four hemispherically rotating gyros mounted with three orthogonal and one slew axis. The AACS flight software selects three gyros as prime and one gyro as a parity detector. When using a foursome that included gyros from both inertial reference units, the fault protection monitor, parity_violation, would be triggered. The software test bed did not have the same problem. The problem was traced to the model. The modeled behavior was that when using a foursome with gyros from each inertial reference unit, the gyro data is sampled from both inertial reference units at the same time, but the internal time tags on the gyro data is either on a 120 msec or 130 msec delta. This produced a timing jitter on the gyro data that caused the parity test to fail. Some of the problems encountered could be fixed, others we had to work around.

The AACS flight software problems detected in the verification program range from a simple code errors, timing errors, to bizarre interaction between response scripts, to hardware design errors. A major timing issue was found on the hardware test bed. A test was run to trigger the repair_bus_controller response script, which acts upon the flight computer. There is only one error monitor that can activate this response, and there is only one path through the response. Therefore, the test results were mystifying. Although the error monitor was triggered, there was absolutely no response. The theory from debugging this test suggested the AACS fault protection software was running out of processing time which prevented the response script from being activated. This issue was invisible on the software test bed since processing time is unconstrained. The amount of run time allocated to the fault protection software in each 125 msec computation cycle was doubled to solve this fundamental problem. Toward the summer of 1997, the problems discovered were ranked as either must-fix or improvement. Only the must-fix items were changed in the launch version of flight software. Of the improvement type FSCs there are on the order of 60 scheduled for a post-launch flight software build.

The AACS fault protection verification test program was very effective in finding flight software errors. There were five to six times as many FSCs written from testing than from analysis or code review. The extensive use of the two test beds demanded rapid correction of test bed errors. The robustness of the flight software as well as the test beds increased due to the verification program.

## 5. BEST PRACTICES

The following guiding principles used in the AACS fault protection verification produced over 450 flight software changes:

- Write test cases as realistic flight scenarios
- Span all the operational modes
- Insert failure during a transition or event on the spacecraft
- Use non-standard hardware configurations
- Change the initial conditions of each test
- Test on hardware test bed as early as possible to find fundamental timing problems
- Ensure fault protection testing is scheduled for a reasonable percentage of the available hardware test bed time. (
- Ensure rapid feedback from finding a bug to the fix to a new delivery
- Review results with code designers to facilitate efficient communications

## 6. SUMMARY

Testing complex fault protection design requires realistic modeling of the system, faster than real-time processing, and a useable fault injection system. The more the testers know about the system, the more bugs they will find. Due to the Cassini AACS fault protection verification program, there is confidence that the fault protection software works well and is robust.

## ACKNOWLEDGMENT

This work was performed at the Jet Propulsion Laboratory, California Institute of Technology, under contract with the National Aeronautics and Space Administration. The author wishes to acknowledge additional people involved in the AACS fault protection verification: Jae Kim, Danny Lam, Kathryn Hilbert, G. Mark Brown, Allan Lee, Garth Watney, Mary Lam, Gurkirpal Singh, and Robert Rasmussen.

## References

[1] John K. Ousterhout, *Tcl and the Tk Toolkit*, Addison-Wesley Publishing Company, Inc., 1994.